Appendix C

Application Programming Interface Examples

APIs (Java style) that can conveniently be used to deal with persistence of objects. None of these APIs assumes knowledge of SQL.

void open(String databaseURL)

Opens the database connection and intializes the data structures as per the Object-Relational Mapping specification corresponding to an ORMId or an ORMFile specified in the databaseURL.

Vector query(String className, String predicate, long maxObjects, long queryFlags, Vector queryDetails)

Returns a list of objects of the given class satisfying the given search condition (predicate). If maxObjects is -1 then all the relevant objects are returned else upto a maximum of maxObjects objects are returned. queryFlags, among other things, specifies if it is a deep query (i.e., all the referenced objects are also retrieved) or shallow query (i.e., just the top-level object is retrieved) Default behavior is deep query. If queryFlag is set for the streaming mode then maxObjects are returned and additional objects may be retrieved using queryFetch() API described below. QueryDetails parameter specifies directed operation options which may be used to control the query in different ways.

Vector queryFetch(long maxObjects, long queryFlags, Vector queryDetails)

Returns a list of upto maxObjects following objects from the object stream opened by a previous call to query(). This is allowed in the same transaction in which the query() call was initiated. QueryDetails parameter specifies directed options which may be used to control the query in different ways.

void queryClose()

Closes the current object stream.

public void insert(Object object, long insertFlags, Vector insertDetails)

Inserts the given object and all its referenced objects that are contained by value. InsertFlags specifies if it is a deep insertion (i.e., all the referenced by value objects are also inserted) or shallow insertion (i.e., just the top-level object is inserted). Default behavior is deep insert insertDetails parameter specifies directed options that may be used to further control the insert operation in different ways.

public void update(Object object, long updateFlags, Vector updateDetails)

Updates the given object and all its referenced objects which are contained by value. The default update semantics are such that the existing persistent copy of the object in the database is replaced by the new updated object. updateFlags

specifies if it is a deep update (i. e., all the referenced by value objects are also updated) or shallow update (i. e., just the top-level object is updated). Default behavior is deep update. updateDetails parameter specifies directed options which may be used to further control the update operation in different ways.

public long update2(String className, String predicate, Vector newValues, long updateFlags)

Updates all objects of the given class satisfying the given search condition (predicate) as per the new attribute values. NewValues parameter is a vector of names and values of updated attributes. (Only shallow update in this case). Returns the number of updated objects.

public void delete(Object object, long deleteFlags, Vector deleteDetails)

Deletes the given object and all its referenced objects which are contained by value. deleteFlags specifies if it is a deep delete (i. e., all the referenced by value objects are also deleted) or shallow delete (i. e., just the top-level object is deleted). Default behavior is deep delete.

public long delete2(String className, String predicate, long deleteFlags)

Deletes all objects of the given class satisfying the given search condition (predicate). deleteFlags specifies if it is a deep delete (i. e., all the referenced by value objects are also deleted) or shallow delete (i. e., just the top-level object is

deleted). Default behavior is deep delete. Returns the number of deleted objects.

public long getNextSequence(String sequenceName, long increment)
throws RemoteException, JDXException;

Returns the next available sequence number for the given sequenceName. The persistent sequence number in the database is incremented by `increment' such that the calling application can safely assume that no other application would get the next sequence number in the range of (returned sequence number n, n+increment-1) both inclusive. In other words, this range (n, n+increment-1) presents a unique set of numbers with respect to the given sequenceName across all applications accessing the same RDBMS. These sequence numbers can typically be used to assign unique ids to different objects.